~~1. A method of improving performance efficiency in a communication system having a plurality of layered protocols by decreasing actual computation, communication latency and layering overhead, the method comprising the steps of:~~

~~determining at least one common execution path in the communication system by identifying a common sequence of operations occurring in the plurality of layered protocols and identifying at least one condition allowing an event to be executed along the common execution path; and~~

~~optimizing the speed of execution of the event along the common execution path by extracting the source code corresponding to the condition and the common sequence of operations, eliminating intermediate data structures and inlining functions called from the common sequence of operations.~~

~~2~~. A method of improving performance efficiency in a communication system having a plurality of layered protocols in a protocol stack, the method comprising the steps of:

determining at least one common execution path in the protocol stack by identifying a common sequence of operations occurring in the plurality of layered protocols;

identifying at least one condition allowing an event to be executed along the common execution path;

path;

extracting source code corresponding to the common sequence of operations; and

modifying the extracted source code corresponding to the common sequence of operations to create modified source code by eliminating source code calling for creation of intermediate data structures from the extracted source code to improve performance efficiency of the communication system.

3. The method of claim 2, further comprising inlining functions called from the modified source code.

4. The method of claim 3, further comprising inserting code, which marks a message as having been compressed, into the modified source code.

5. The method of claim 4, further comprising inserting code, which stores a reformatting function with the compressed message, into the modified source code.

6. The method of claim 2, further comprising inserting code, which will compress a header, into the modified source code.

7. The method of claim 6, wherein the header is an addressing header.

8. The method of claim 7, wherein the addressing header is compressed by using a connection identifier.

9. The method of claim 6, wherein the header is a constant header.

10. The method of claim 9, wherein a multiplexing index is used to multiplex a plurality of virtual channels over a single channel.

11. The method of claim 2, further comprising inserting code, which will reformat a message by functions which regenerate constant fields and move non-constant fields to a normal location in the message, into the modified source code.

12. The method of claim 2, wherein the modified source code is further modified so that at least one of the common sequence of operations is performed after a message corresponding to the event has been transmitted.

13. The method of claim 12, wherein the at least one of the common sequence of operations includes an operation which buffers a message.

14. The method of claim 2, wherein the modified source code is further modified so that at least one of the common sequence of operations is performed after a message corresponding to the event has been delivered.

15. The method of claim 2, wherein the intermediate data structures are event records designed to pass information between protocol layers in the protocol stack.

16. A method for processing an event comprising:

receiving an event at an event processor, the event processor having a trace handler and a protocol stack, the protocol stack having a plurality of protocol layers;

determining whether the event satisfies a trace condition;

if the event does not satisfy the trace condition, sending the event to the protocol stack and processing the event using the protocol layers, the protocol layers executing a first kind of operations and a second kind of operations;

if the event satisfies the trace condition, sending the event to the trace handler and processing the event using the trace handler, the trace handler executing the second kind of operations, but not executing the first kind of operations.

17. The method of claim 16, wherein the second kind of operations include protocol operations.

18. The method of claim 17, wherein the protocol operations include protocol operations which would be executed by the protocol stack if an event which satisfies the trace condition were sent to the protocol stack.

19. The method of claim 16, wherein the first kind of operations include the use of event records to pass information between protocol layers in the protocol stack.

20. The method of claim 16, wherein the second kind of operations executed by the trace handler are directed by computer readable code which has inlined functions, the inlined functions being functions which would be called

by the protocol stack if an event satisfying the trace condition were sent to the protocol stack.

21. The method of claim 16, wherein the second kind of operations executed by the trace handler achieve results which would be achieved by the protocol stack if an event which satisfies the trace condition were sent to the protocol stack.

22. The method of claim 16, wherein at least one of the first kind of operations is executed, with respect to a message corresponding to an event which satisfies the trace condition, after the message has been transmitted.

23. The method of claim 22, wherein the at least one of the first kind of operations includes buffering the message.

24. The method of claim 22, wherein at least one of the first kind of operations is executed, with respect to a message corresponding to an event which satisfies the trace condition, after the message has been delivered.

25. The method of claim 16, wherein the second kind of operations executed by the trace handler are directed by computer readable code which has inlined functions, the inlined functions being functions which would be called by the protocol stack if an event satisfying the trace condition were sent to the protocol stack.

26. The method of claim 16, wherein the trace handler executes additional operations which mark a message, the message corresponding to an event which satisfies the trace condition, with a marker which indicates whether

compression operations have been performed in conjunction with processing the event which satisfies the trace condition.

27. The method of claim 16, wherein the trace handler executes additional operations which store a reformatting function with a compressed message that corresponds to an event which satisfies the trace condition.

28. The method of claim 16, wherein the trace handler executes additional operations which compress an addressing header of a message that corresponds to an event which satisfies the trace condition.

29. The method of claim 28, wherein a connection identifier is used to effect compression of the addressing header.

30. The method of claim 16, wherein the trace handler executes additional operations which use a multiplexing index to create a virtual channel for sending a message that corresponds to an event which satisfies the trace condition.

31. The method of claim 16, wherein the trace handler executes additional operations which compress a constant header of a message that corresponds to an event which satisfies the trace condition.

32. The method of claim 16, wherein the first kind of operations include a layering operation.

33. The method of claim 16, further comprising reformatting a message to generate a constant field prior to executing the second kind of operations.

34. The method of claim 16, further comprising moving a non-constant field to a normal location in a message prior to executing the second kind of operations.